PERGAMON Computers and Mathematics with Applications 39 (2000) 77–88

An International Journal
computers & mathematics
with applications

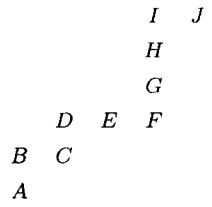
www.elsevier.nl/locate/camwa

Unimodular Arrays

E. BERLEKAMP
University of California, Berkeley, CA 94720, U.S.A.
berlek@math.berkeley.edu

Abstract—This paper explores relationships between optimal erasure-burst-correcting convolutional codes, arrays of integers within which all submatrices whose upper left corner lies on a certain boundary have determinant one, and enumerations of certain parts within such arrays. © 2000 Elsevier Science Ltd. All rights reserved.

A "duke"[1] is a chessman whose capabilities are the intersection of the king and the rook: it can move one square in any horizontal or vertical direction. In this paper, I am concerned only with a "monotonic duke", which can move one square north or one square east. Here is a typical path of such a chesspiece, with successive locations denoted by the letters $A, B, C, \ldots$:



A monotonic duke's path.

Such a path may be taken as the boundary of some region of the chessboard that lies southeast of it. Let us assign the number "1" to each of the boundary locations. We then seek to assign other integers to the locations on the chessboard that lie southeast of this path, in such a way that every submatrix whose upper left corner lies on the boundary has value 1.[2] In the present example, this yields the following:

			1	1
			1	2
			1	3
	1	1	1	4
1	1	2	3	13
1	2	5	9	x

Array 1. A unimodular array, bounded by the monotonic duke's path.

[1] It has been suggested that "cook" might be a better pneumonic intersection of "king" and "rook".

[2] Arrays with monotonic boundaries arise naturally in the theory of convolutional codes. Requiring all square matrices whose corners lie on the boundary to be nonsingular is necessary and sufficient to ensure the capability to correct every erasure burst with minimum delay. See [1,2].

0898-1221/00/$ - see front matter © 2000 Elsevier Science Ltd. All rights reserved. Typeset by AMS-TEX
PII: S0898-1221(00)00111-5

$$1 = \begin{vmatrix} 1 & 1 \\ 1 & 2 \end{vmatrix} = \begin{vmatrix} 1 & 2 \\ 1 & 3 \end{vmatrix} = \begin{vmatrix} 1 & 3 \\ 1 & 4 \end{vmatrix} = \begin{vmatrix} 1 & 1 \\ 2 & 3 \end{vmatrix} = \begin{vmatrix} 1 & 2 \\ 2 & 5 \end{vmatrix} = \begin{vmatrix} 1 & 4 \\ 3 & 13 \end{vmatrix}$$
$$= \begin{vmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 2 & 5 & 9 \end{vmatrix} \stackrel{?}{=} \begin{vmatrix} 1 & 1 & 4 \\ 1 & 2 & 13 \\ 5 & 9 & x \end{vmatrix}.$$

We now seek to assign the value of x in such a way as to control the value of the determinant of the square matrix whose main diagonal runs from the boundary to x:

$$\begin{vmatrix} 1 & 1 & 4 \\ 2 & 3 & 13 \\ 5 & 9 & x \end{vmatrix} \stackrel{?}{=} 1.$$

Expanding this determinant by cofactors of the last column yields

$$\begin{vmatrix} 1 & 1 & 4 \\ 2 & 3 & 13 \\ 5 & 9 & x \end{vmatrix} = 4 \begin{vmatrix} 2 & 3 \\ 5 & 9 \end{vmatrix} - 13 \begin{vmatrix} 1 & 1 \\ 5 & 9 \end{vmatrix} + x \begin{vmatrix} 1 & 1 \\ 2 & 3 \end{vmatrix}.$$

Since the cofactor of x is a determinant that is already known to be 1, it becomes apparent that our equation for x must have a unique solution, and the solution must be an integer. In the present example, this gives

$$1 = 4 \cdot 3 - 13 \cdot 4 + x \cdot 1,$$
$$x = 41.$$

Evidently, the same argument ensures that *any* monotonic boundary determines a unique unimodular array to its southeast. Here are some examples that might occur on a sufficiently large chessboard:

							1
						1	1
					1	1	1
				1	1	2	5
			1	1	2	5	14
		1	1	2	5	14	42
	1	1	2	5	14	42	132
1	1	2	5	14	42	132	429

Array 2. Linear boundary with slope 1.

								1	1
						1	1	1	2
				1	1	1	2	3	7
		1	1	1	2	3	7	12	30
	1	1	1	2	3	7	12	30	55	143
1	1	1	2	3	7	12	30	55	143	273	728

Array 3. Linear boundary with slope 1/2.

														1
										1	1	1	1	
						1	1	1	1	2	3	4		
				1	1	1	1	2	3	4	9	15	22	
			1	1	1	1	2	3	4	9	15	22	52	91	140
1	1	1	1	2	3	4	9	15	22	52	91	140	340	612	969

Array 4. Linear boundary with slope 1/3.

					1
					1
				1	1
				1	2
			1	1	3
			1	2	7
		1	1	3	12
		1	2	7	30
	1	1	3	12	53
	1	2	7	30	143
1	1	3	12	55	273
1	2	7	30	143	728

Array 5. Linear boundary with slope 2 (transpose of Array 3).

					1
					1
					1
				1	1
				1	2
				1	3
			1	1	4
			1	2	9
			1	3	15
		1	1	4	22
		1	2	9	52
		1	3	15	91
	1	1	4	22	140
	1	2	9	52	340
	1	3	15	91	612
1	1	4	22	140	969

Array 6. Linear boundary with slope 3 (transpose of Array 4).

1	1	1	1	1	1	1	1	1	1
1	2	3	4	5	6	7	8	9	10
1	3	6	10	15	21	28	36	45	55
1	4	10	20	35	56	84	120	165	220
1	5	15	35	70	126	210	330	495	715

Array 7. Pascal's triangle.

										1
										1
									1	1
								1	1	2
								1	2	5
							1	1	3	9
						1	1	2	7	32
						1	2	5	19	66
					1	1	3	9	37	136
				1	1	2	7	23	99	377
				1	2	5	19	66	293	1144
			1	1	3	9	37	136	624	2504
		1	1	2	7	23	99	377	1768	7229
		1	2	5	19	66	293	1144	5452	22610
	1	1	3	9	37	136	624	2504	12151	51211
1	1	2	7	23	99	377	1768	7229	35530	?

Array 8. Linear boundary with slope 3/2.

DETERMINING UNIMODULAR ARRAYS BY VECTOR SHIFT-AND-ADD OPERATIONS

Let y denote some starting point on the boundary, and let x denote some finishing point on the boundary. Then our monotonic duke can travel from y to x by following the boundary. In most cases, he can also take alternate paths from y to x by traveling through other locations on the chessboard. We say that such a path is *constrained* if it avoids trespassing on any locations northwest of the boundary.

THEOREM. *The value of the label at the location of the unimodular array that lies on the same row as y and the same column as x is equal to the number of constrained paths from y to x.*

Before proving this theorem, let us examine one of its major consequences.

ALGORITHM. *The number of constrained paths from y to each of the locations on a diagonal, $\mathbf{d}$, may be determined by a shift-and-add operation applied to the vector whose elements give the number of constrained paths from y to each location on the previous diagonal.*

The proof that this algorithm works is an obvious consequence of the fact that a monotonic duke can enter a square in only two ways: from the west or from the south. Both of these locations are on the previous diagonal.

Let us apply the algorithm to our initial example. Here is the boundary of Array 1, with some additional locations lettered for reference:

			I	J					
			H	a	b				
			G	c	d	e			
	D	E	F	f	g	h	i		
B	C	j	k	l	m	n	o	p	
A	q	r	s	t	u	v	w	x	y
	z	a'	b'	c'					
		d'	e'	f'					
			g'	h'					
				i'					

Here are the numbers of constrained paths from A to the diagonal starting at:

A						1					
B	q					1	1				
C	r					2	1				
D	j	s				2	3	1			
E	k	t				5	4	1			
F	l	u				9	5	1			
G	f	m	v			9	14	6	1		
H	c	g	n	w		9	23	20	7	1	
I	a	d	h	o	x	9	32	43	27	8	1
J	b	e	i	p	y	41	75	70	35	9	1

(endpoints for paths counted in table at right)

Here are the numbers of constrained paths from B to the diagonal starting at:

B					1					
C					1					
D	j				1	1				
E	k				2	1				
F	l				3	1				
G	f	m			3	4	1			
H	c	g	n		3	7	5	1		
I	a	d	h	o	3	10	12	6	1	
J	b	e	i	p	13	22	18	7	1	

(endpoints for paths counted in table at right)

Here are the numbers of constrained paths from D to the diagonal starting at:

D				1					
E				1					
F				1					
G	f			1	1				
H	c	g		1	2	1			
I	a	d	h	1	3	3	1		
J	b	e	i	4	6	4	1		

(endpoints for paths counted in table at right)

Here are the numbers of constrained paths to J from the diagonal starting at:

J					1					
I	a				1	1				
H	c				2	1				
G	f				3	1				
F	l				4	1				
E	k	t			4	5	1			
D	j	s	c'		4	9	6	1		
C	r	b'	f'		13	15	7	1		
B	q	a'	e'	h'	13	28	22	8	1	
A	z	d'	g'	i'	41	50	30	9	1	

(startpoints for paths counted in table at right)

Here are the numbers of constrained paths to I from the diagonal starting at:

I				1				
H				1				
G				1				
F				1				
E	k			1	1			
D	j	s		1	2	1		
C	r	b'		3	3	1		
B	q	a'	e'	3	6	4	1	
A	z	d'	g'	9	10	5	1	

(startpoints for paths counted in table at right)

Here are the numbers of constrained paths to E from the diagonal starting at:

E			1			
D	j		1	1		
C	r		2	1		
B	q	a'	2	3	1	
A	z	d'	5	4	1	

(startpoints for paths counted in table at right)

Here is the matrix of numbers of paths from left ends of rows to the E diagonal:

$\mathcal{FROM} =$

From \ To	E	k	t
E	1	0	0
B	2	1	0
A	5	4	1

Here is the matrix of numbers of paths from the E diagonal to tops of columns:

$\mathcal{TO} =$

From \ To	E	I	J
E	1	1	4
k	0	1	5
t	0	0	1

Here is the matrix of numbers of paths from starts of rows to tops of columns:

$\mathcal{FROM} * \mathcal{TO} =$

From \ To	E	I	J							
D	1	1	4		1	0	0		1	1	4
B	2	3	13	=	2	1	0		0	1	5
A	5	9	41		5	4	1		0	0	1

Here is the unimodular array within which this matrix fits:

			1	1
			1	2
			1	3
	1	**1**	**1**	**4**
1	1	**2**	**3**	**13**
1	2	**5**	**9**	**41**

This example easily generalizes to the following proof of the theorem.

PROOF OF THE THEOREM. If false, let location $[x, y]$ be a minimum counterexample, so that the theorem is true for all other entries in the square matrix whose diagonal runs from the boundary to location $[x, y]$.

Consider the matrix whose left corner lies on the boundary, whose lower right corner lies at location $[x, y]$, and whose main diagonal is $\mathbf{d}$. Construct two matrices, $\mathcal{TO}$ and $\mathcal{FROM}$. The generic entry of $\mathcal{TO}$ is the number of constrained paths from a boundary point at the left end of a row of the array to a point on the diagonal $\mathbf{d}$. The generic entry of $\mathcal{FROM}$ is the number of constrained paths from a point on the diagonal $\mathbf{d}$ to a boundary point at the top of a column of the array. Since constrained paths are monotonic, both the $\mathcal{TO}$ and $\mathcal{FROM}$ matrices are triangular, with ones on their diagonals. It is clear that $\mathcal{TO}$ and $\mathcal{FROM}$ are both unimodular.

Any constrained path from a boundary point below the diagonal to a boundary point above the diagonal must cross the diagonal exactly once. Therefore, the number of such paths may be found by summing the paths through each point on the diagonal; the number of paths through any particular diagonal point is the product of the corresponding entries in the $\mathcal{TO}$ and $\mathcal{FROM}$ matrices. Therefore, the product of the $\mathcal{TO}$ and $\mathcal{FROM}$ matrices is the matrix whose elements count the number of constrained paths. ∎

Notice that the theorem not only eliminates the need to compute cofactors of determinants, but it also provides a factorization of any relevant matrix into a product of two triangular matrices which are more easily inverted.

Even though the algorithm requires only vector shift-and-add instructions, a simple formula is even more desirable. For some boundaries, such as Array 7, the results of the shift-and-add recursions are well known. We now generalize this to a broader class of boundaries, which are particularly important in the study of erasure-burst-correcting convolutional codes.

LINEAR BOUNDARIES

The array corresponding to a convolutional code of rate R is, to within roundoff errors, a straight line of slope $(1 - R)/R$. Here are the rates corresponding to our prior examples:

Array	Slope	Code Rate
1	irreg	irreg
2	1	1/2
3	1/2	2/3
4	1/3	3/4
5	2/1	1/3
6	3/1	1/4
7	irreg	irreg
8	3/2	2/5

There is an obvious duality between the arrays corresponding to rate R and to rate $1 - R$. Their boundaries have reciprocal slopes and the arrays can be transformed into each other by reflecting about a diagonal of slope -1.

The most popular code rates are of the form $k/(k + 1)$. These correspond to arrays with boundaries that move up 1, right k, up 1, right k, up 1, right k, ... (see Figure 1).

For purposes of discussion, let y denote a starting point on the boundary, and we let the Greek letters $\alpha, \beta, \ldots, \omega$ denote certain key locations *above* the boundary (see Figure 2).

We now consider the total number of unconstrained paths from y to x. Since each move along any such path increments the diagonal, it is clear that all paths have the same length n. If the path is unconstrained, its j "up" moves may be chosen arbitrarily, and so the number of such

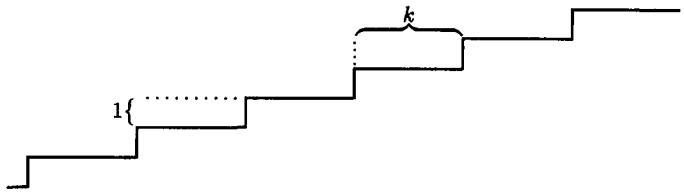


Figure 1.

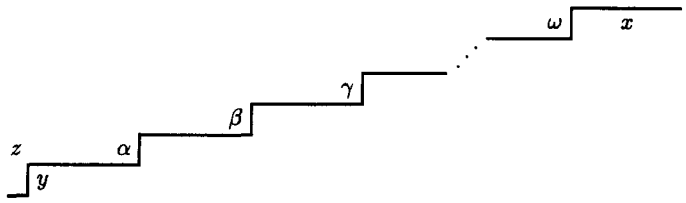


Figure 2.

paths is the binomial coefficient,

$$\binom{n}{j}.$$

Let us also consider the point z, which lies diagonally northwest of y. Its duke-distance from x is also n, and the number of (unconstrained) paths from z to x is

$$\binom{n}{j-1}.$$

We now observe that the number of unconstrained paths from y to x is equal to the number of constrained paths from y to x plus the total number of paths from y to x that trespass into the region northwest of the boundary. It is easily seen that the last trespassing location of any trespassing path must be one of the locations we have labeled with a Greek letter. The portion of the path *after* the last trespassing location must be constrained.

For each such Greek letter, say γ, the number of unconstrained paths from z to γ is

$$\binom{ki+i-1}{i-1}.$$

The number of unconstrained paths from y to γ is

$$\binom{ki+i-1}{i} = k\binom{ki+i-1}{i-1}.$$

In other words, for *every* choice of Greek letter γ, the number of unconstrained paths from y to γ is exactly k times of the number of unconstrained paths from z to γ. So, evidently, the number of trespassing paths from y to x is precisely k times the number of unconstrained paths from z to x. Therefore, the number of unconstrained paths from y to x is

$$\binom{n}{j} - k\binom{kj+j-1}{j-1}.$$

The values of Arrays 2–6 agree with this formula. Here are the detailed formulas for Array 2, the Catalan numbers $(1/(n+1))\binom{2n}{n}$:

$$1 = \binom{0}{0},$$

$$1 = \binom{1}{0} \qquad = \binom{2}{1} - \binom{2}{0},$$

$$
\begin{aligned}
2 &= \binom{3}{1} - \binom{3}{0} && = \binom{4}{2} - \binom{4}{1},\\
5 &= \binom{5}{2} - \binom{5}{1} && = \binom{6}{3} - \binom{6}{2},\\
14 &= \binom{7}{3} - \binom{7}{2} && = \binom{8}{4} - \binom{8}{3},\\
42 &= \binom{9}{4} - \binom{9}{3} && = \binom{10}{5} - \binom{10}{4},\\
132 &= \binom{11}{5} - \binom{11}{4} && = \binom{12}{6} - \binom{12}{5},\\
429 &= \binom{13}{6} - \binom{13}{5} && = \binom{14}{7} - \binom{14}{5}.
\end{aligned}
$$

Here are the formulas for Arrays 3 and 5:

$$
\begin{aligned}
1 &= \binom{0}{0},\\
1 &= \binom{1}{0},\\
1 &= \binom{2}{0} && = \binom{3}{1} - 2\binom{3}{0},\\
2 &= \binom{4}{1} - 2\binom{4}{0},\\
3 &= \binom{5}{1} - 2\binom{5}{0} && = \binom{6}{2} - 2\binom{6}{1},\\
7 &= \binom{7}{2} - 2\binom{7}{1},\\
12 &= \binom{8}{2} - 2\binom{8}{1} && = \binom{9}{3} - 2\binom{9}{2},\\
30 &= \binom{10}{3} - 2\binom{10}{2},\\
53 &= \binom{11}{3} - 2\binom{11}{2} && = \binom{12}{4} - 2\binom{12}{3},\\
143 &= \binom{13}{4} - 2\binom{13}{3},\\
273 &= \binom{14}{4} - 2\binom{14}{3} && = \binom{15}{5} - 2\binom{15}{4},\\
728 &= \binom{16}{5} - 2\binom{16}{4}.
\end{aligned}
$$

Here are the formulas for Arrays 4 and 6:

$$
\begin{aligned}
1 &= \binom{3}{0} && = \binom{4}{1} - 3\binom{4}{0},\\
2 &= \binom{6}{1} - 3\binom{5}{0},\\
3 &= \binom{6}{1} - 3\binom{6}{0},\\
4 &= \binom{7}{1} - 3\binom{7}{0} && = \binom{8}{2} - 3\binom{8}{1},
\end{aligned}
$$

$$
\begin{aligned}
9 &= \binom{9}{2} - 3\binom{9}{1},\\
15 &= \binom{10}{2} - 3\binom{10}{1},\\
22 &= \binom{11}{2} - 3\binom{11}{1} = \binom{12}{3} - 3\binom{12}{2},\\
52 &= \binom{13}{3} - 3\binom{13}{2},\\
91 &= \binom{14}{3} - 3\binom{14}{2},\\
140 &= \binom{15}{3} - 3\binom{15}{2} = \binom{16}{4} - 3\binom{16}{3}.
\end{aligned}
$$

Notice that the boundary for Array 4 has period 4, with one move upward and three moves rightward. These integers, 1 and 3, are the constant multipliers of the binomial coefficients that appear in the formulas.

Although we have no proof, we empirically observe that somewhat similar formulas hold for Array 8, whose boundary has period 5, with two moves upward and three moves rightward:

$$
\begin{aligned}
1 &= -2\binom{1}{1} + 3\binom{1}{0},\\
1 &= 2\binom{2}{1} - 3\binom{2}{0},\\
1+2 &= 2\binom{3}{1} - 3\binom{3}{0},\\
2+3 &= 2\binom{4}{1} - 3\binom{4}{0},\\
5 &= -2\binom{6}{3} + 3\binom{6}{2},\\
7 &= 2\binom{7}{3} - 3\binom{7}{2},\\
9+19 &= 2\binom{8}{3} - 3\binom{8}{2},\\
23+37 &= 2\binom{9}{3} - 3\binom{9}{2},\\
66 &= -2\binom{11}{5} + 3\binom{11}{4},\\
99 &= 2\binom{12}{5} - 3\binom{12}{4},\\
136+293 &= 2\binom{13}{5} - 3\binom{13}{4},\\
377+624 &= 2\binom{14}{5} - 3\binom{14}{4},\\
1144 &= -2\binom{16}{7} + 3\binom{16}{6},\\
1768 &= 2\binom{17}{7} - 3\binom{17}{6},\\
2504+5452 &= 2\binom{18}{7} - 3\binom{18}{6},
\end{aligned}
$$

$$7229 + 12151 = 2\binom{19}{7} - 3\binom{19}{6},$$
$$22610 = -2\binom{21}{9} + 3\binom{21}{8},$$
$$35530 = 2\binom{22}{9} - 3\binom{22}{8}.$$

The patterns are clear, but I know no explanation. Why does the formula apply to an individual entry, then to sums of pairs of entries from different rows, and then to the *negative* of an entry?

I have not found any simple formulas for the individual entries within each pair which is summed together in the above formulas.

COPY-ZERO AND ZERO-COPY ALGORITHM

The values of the coefficients of the unimodular arrays must be reduced modulo 2 in order to become the generator sequences for binary convolutional codes. Indeed, the original investigation of erasure-burst-correcting convolutional codes [1] discovered some interesting theorems without realizing the close relationship to the enumerative combinatorics presented in the present paper.

By applying Lucas' Theorem [2], one can obtain simple formulas for the binomial coefficients reduced modulo any prime p. If one reduces the semi-infinite sequence along any row of Array 3 modulo 2, one obtains the following sequence of bits:

11101100111000001110110000000000111011....

This sequence can be constructed by the following very fast algorithm.

We first write "ones" to correspond to the part of the row that is on the boundary.

11

Then we alternately "copy" and "zero" each bit of the prior sequence.

10

This new sequence is then appended to the former sequence.

1110

We are now ready to double the length of the sequence again, although this time we alternately "copy-2" and "zero-2".

1100

This new sequence is then appended to the former sequence.

11101100

We are now ready to double the length of the sequence again, although this time we alternately "copy-4" and "zero-4".

11100000

On the next doubling of the sequence, we "copy-8" and "zero-8"; then "copy-16" and "zero-16"; etc.

....

This "copy-zero" algorithm works for all linear boundaries corresponding to rates of the form $k/(k+1)$, where k is even. If k is odd, then the "copy-zero" algorithm becomes the "zero-copy" algorithm. In the simple case of rate 1/2, this gives a very low density generator sequence:

11010001000000010000000000000001....

This sequence corresponds to the fact that the n^{th} Catalan number is odd only if n is a power of 2.

Nothing comparable to the "copy-zero" algorithm is known for boundaries such as Array 8, which corresponds to the code of rate 2/5.

REFERENCES

1. E. Berlekamp, A class of convolution codes, *Information & Control* **6**, 1–13, (1963).
2. E. Berlekamp, *Algebraic Coding Theory*, (see p. 113 for Lucas' Theorem on binomial coefficients mod p), Aegean Park Press, (1984).